Factors to Consider When Implementing Automated Software Testing

By Larry Yang, MBA, SSCP, Security+, Oracle DBA OCA, ASTQB CTFL, ITIL V3 ITM

Testing is a major component of the Software Development Lifecycle (SDLC), constituting a prominent cost driver for both government and industry entities. Therefore, many businesses are automating their software testing in order to save money and improve quality. When considering whether automation is a viable option, businesses must take several factors into account. The purpose of this document is to illuminate these factors.

Software development and integration is a continuous process throughout the acquisition life cycle. Automated Software Testing can improve testing capabilities, replacing some of the resource-intensive manual efforts, and can be executed alone or in conjunction with manual testing. Regardless of the complexity, technical maturity, or requirements of the software program, Automated Software Testing may be a viable option.

ADVANTAGES AND DISADVANTAGES OF AUTOMATED SOFTWARE TESTING

Automated Software Testing is characterized by the following advantages and disadvantages:

- Advantages
    1. Saves Time and Money
        - Reduces setup time by automating data staging.
        - Streamlines regression testing after every software modification, patch or release by reducing manual labor.
        - Reduces overall test execution times at minimal cost.
    2. Improves Software Quality
        - Minimizes human errors.
        - Enhances repeatability and consistency.
        - Reduces the risk of errors through more thorough testing (more detail below).
    3. Expands and Enhances Test Coverage
        - Automated Software Testing can execute thousands of complex test cases throughout each test run.
        - Lengthy tests can be run unattended, e.g., overnight.
    4. Allows additional testing not suitable to Manual Software Testing
        - Load and performance testing
            - Simulation of thousands of users and concurrent processes.
        - Endurance testing

        - Examination of a software system as it runs for a long duration, and measurement of the system's reaction parameters.
    - Longevity testing
        - Evaluates a system's ability to handle a constant, moderate work load for a long time.
    - Race-condition testing
        - Able to simulate complex scenarios caused by timing issues that occur only when a system is handling heavy volumes from multiple connections.
        - Automated testing allows identification of potentially catastrophic issues early, before they cause system outages.
    - Large and greatly varied data sets that the automated tool is able to generate, allowing more exhaustive testing
- Disadvantages
    1. Challenging process which requires appropriate resources.
        - Automation Architect and Automation Engineers (often costly and difficult to find).
        - Developers with knowledge of scripting and programming, e.g., Java or Visual Basic.
        - Subject Matter Experts (SME) with firm grasp of application being automated.
    2. Additional costs for setup (e.g., purchase of the automation tool, training of staff, and maintenance of test scripts), equipment, lab, and annual software maintenance fees.
    3. Errors in automated scripts, though rare, can negate data collected and force re-testing.

SCOPE OF THE INITIATIVE FOR AUTOMATED SOFTWARE TESTING

Once the advantages and disadvantages have been weighed, the scope of the initiative must be defined. The software application(s) should be analyzed to determine which portions can be automated. Good candidates are test cases that:

- Process a large amount of data.
- Are time-consuming or repetitious.
- Are complex or difficult to execute.
- Are prone to human error during manual testing.
- Perform common, frequently executed processes.
- Assess functionality of core and critical business functions.
- Ensure continued software functionality through regression testing.

CHECKLIST FOR AUTOMATED SOFTWARE TESTING

The analysis above will define whether your application has a large enough scope to justify Automated Software Testing. If so, the following checklist should be used.

- Leadership Approval – Must be established during project planning phase to ensure support.
- Calculation of Business Benefits – Perform detailed analysis to predict value of implementation

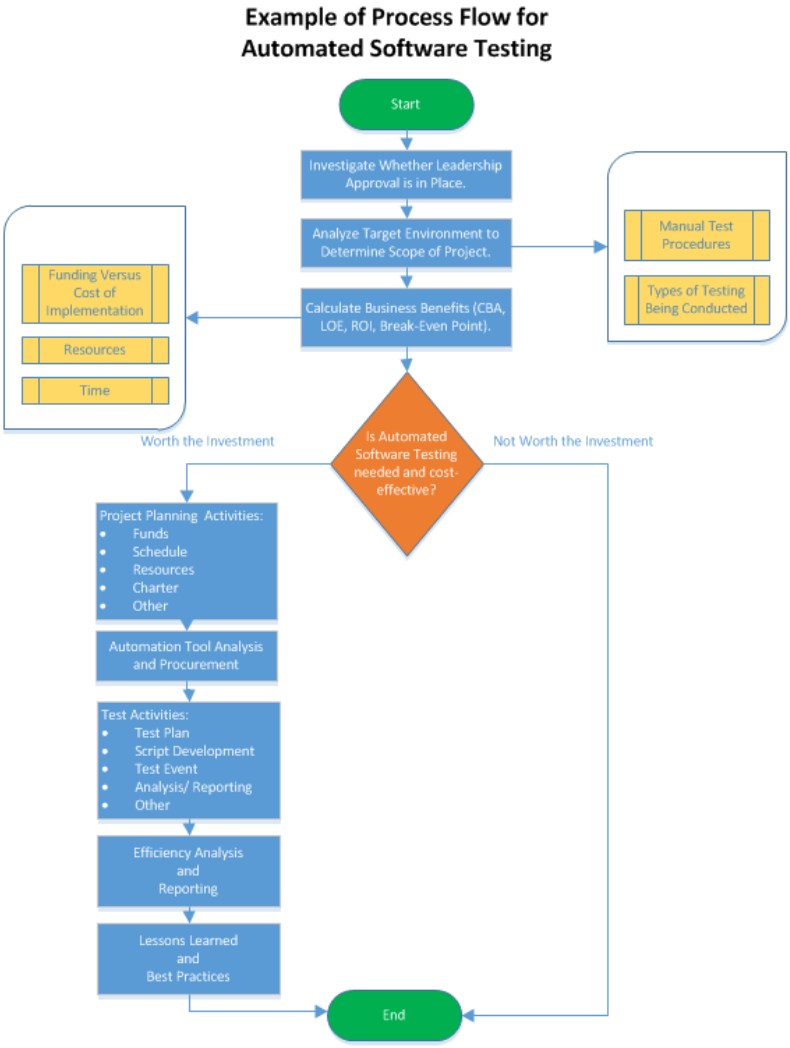
1. Cost-Benefit Analysis (CBA) or Business Case Analysis (BCA): Perform thorough business study to show the costs and benefits of Automated Software Testing.
2. Level of Effort (LOE): Investigate the existing LOE for the project to prevent any issues, e.g., concurrent test environment usage resulting in resource collisions.
3. Return on Investment (ROI): Run a calculated analysis of automating your program to determine the total cost, savings per year, benefits per year, and overall gain (benefit divided by investment).
4. Break-even point: Determine when the program will recoup the costs.

- Tools – Decide which automated tool(s) to use for your program. Research all possible tool options, list the pros and cons of each tool and decide which one will benefit your program the most. Considerations include:
    1. Compatibility with system being automated
    2. Browsers and operating systems supported
    3. Applications supported, e.g., mainframe, back-end, and web
    4. Programming languages supported
    5. Level of coding experience required among Automation Engineers
    6. Amount of coding required to generate automated test scripts
    7. Reusability of test components
    8. Load testing capability
    9. Reporting and analysis capabilities
    10. Ability to record and playback
    11. User-friendliness of interface
    12. Costs
        - Initial cost: Procuring the software/tool
        - Maintenance costs: Tool support including critical updates, patches, bug fixes
        - Support costs: Additional fees, if applicable
        - Licensing costs: Number needed for team members plus a few extras for backup
        - Training costs: Classes needed for the team, if applicable
- Environment – Define needs of development and testing environment
    1. Ideal development setting will simulate a production-like environment.
    2. Decide which data sets are needed for automated testing. Stage data prior to test execution if needed.
    3. Ensure all development and testing accounts are set up, with appropriate permissions.
    4. Establish a process for software and data baselining.
- Location – Determine the requirements for development and testing site
    1. Consider whether project requires a lab, conference room, or both, and whether it should be located in-house or an external facility.
    2. Ensure space is adequate to support team, observers, meetings, and equipment.
    3. Weigh connectivity requirements – security level, wireless vs. wired network.
    4. Compare costs of various options to ascertain best fit.
- Automated Software Testing Team – Determine and fill resource requirements.

1. Develop Knowledge, Skills, and Abilities (KSA) required (e.g., Test and Evaluation).
2. Analyze programming skills needed (e.g., Java, C, C++, Visual Basic).
3. Compose team – testers, software engineers, database administrators (DBA), SME, etc.
4. Internal versus external resources: Weigh costs, availability and long-term benefits. Determine whether existing program resources are available to assist.
5. Create position descriptions and complete necessary hiring actions, e.g., online job postings.

- Overall Costs to Implement – Quantify funding needed for implementation.
    1. Automated software tool: Initial, maintenance, support, licensing, and training
    2. Training: Program-specific training and automation-tool training
    3. Labor: Number of people, length of time needed and billing rate
    4. Location and facility: Development and testing site
    5. Contract: Fees associated with tool procurement, contractor support, facility
    6. Sustainment and maintenance: Automated test-script updates, repository, documentation
- Prototype – Develop and present to ensure Proof of Concept.
- Documentation – Written by the team to ensure a successful implementation.
    1. Project Management Documents
        - Project Structure, Monitoring & Control– business alignment, project initiation and acceptance
        - Project Management Plan (PMP) – scope, schedule, cost and performance
        - Project Charter – mission, objectives, deliverables and member expectations
        - Integrated Master Schedule (IMS) – resource-loaded timeline with milestones
        - Communications Plan – details of stakeholder engagement
        - Data/Reporting Plan – consistent reporting procedure
        - Work Breakdown Structure (WBS) and WBS Dictionary – hierarchy of work elements
        - Organizational Breakdown Structure (OBS) – tiered depiction work teams/functions
        - Financial Structure of Project –billing elements and network activities
        - Procurement Strategy – specific procurement actions for services or supplies
        - Project Spend Plan – program-level budget development and execution
        - Project Resource Plan – rough order of magnitude (ROM) estimates of manpower, facilities, IT and other operational requirements
        - Project Closeout Plan – facilities, materials, contract, financial and workforce
    2. Technical Execution
        - Systems Engineering Plan (SEP) – guide for all technical aspects of the program
        - High-Level System Requirements – top-level requirements, e.g., stakeholder
        - Decomposed System Requirements – breakdown of high-level requirements
        - Requirements Traceability Matrix (RTM) – linking of requirements throughout validation process
        - System Design – engineering design overview for Automated Software Testing
        - Technical Review Action Plan (TRAP) – guide for facilitating Systems Engineering Technical Review (SETR) event, e.g., entry/exit criteria and approval process

    - Automation and Development Guide – coding standards, script-development process, configuration instructions
    - Automated Test Script Suite – developed by test team from manual procedures
    - Detailed Test Plan (DTP) - overall Test and Evaluation structure and objectives
    - Test Report – results of testing to include results and analysis of testing
    - Best Practices & Lessons Learned
- Other Considerations – Must be taken into account when deciding whether to pursue Automated Software Testing.
    1. Industry Research: Find out sources of best practices, tips, tool suggestions, and support
    2. Efficiency: Develop a strategy for how Automated Software Testing will maximize efficiency.
    3. Training: After you choose the tool, decide whether or not training classes or other tools (books, tutorials, guides) are needed for the team. Training classes are a major expense.
    4. Process Flow: Create a chart to show how you will automate your program (see below).



CONCLUSION

This paper outlines the factors to evaluate and the process to follow in implementing Automated Software Testing. Initial factors must be present such as leadership approval, funding, and resources. Program managers should evaluate the environment to determine whether it meets the scope for an implementation, and if so, follow the detailed project checklist for maximum success.

In conclusion, even though Automated Software Testing requires up-front costs, it also saves money and improves quality, along with other benefits. For programs that prove to be good candidates, the benefits far outweigh the costs.

With over 12 years in Information Technology and experience in Automated Software Testing, Larry Yang has both formal education (MBA, BS in Computer Science and AS in Computer Programming) and technical credentials (SSCP, Security+, Oracle DBA Certified Associate, ITIL v3 Intermediate, and ASTQB Certified Tester Foundation Level).

Mr. Yang is the Project Lead of the United States Marine Corps / Special Operations Command (USMC/SOCOM) Portfolio Test Automation Project at SPAWAR Systems Center Atlantic. He received funding from Chief of Naval Operations Innovation, Technology Requirements, and Test and Evaluation (OPNAV N84) in support of the U.S. Navy Automated Test and Analysis (ATA) Program for FY16.

The project chose an existing software system as a pilot for proving the automated testing solution and capability. The Major Automated Information System (MAIS) Acquisition Category I (ACAT-I) program was selected due to its complexity, requirements, technical maturity, and interface count. The project implemented Automated Software Testing for this system, and in the process captured metrics to ensure that the time and money had been well spent. The checklist in this paper is what Mr. Yang developed to guide the Test Automation Project.